Mar 28, 2024

ZAN

Security Assessment

BEVMSwap

—

Professional Service

Table of Contents

1. Overview

1.1. Executive Summary

BEVMSwap is a decentralized exchange(DEX) built on BEVM, a BTC Layer2. This report has been prepared for **BEVMSwap** project to discover issues and vulnerabilities in the source code of this project as well as any contract dependencies that were not part of an officially recognized library.

Conducted by Static Analysis, Formal Verification and Manual Review, we have identified **1 Medium and 7 Informational issues**.

The project team has **fixed the Medium security vulnerability and adopted all optimization suggestions except for I-04** in commit 2da7fad.

1.2. Project Summary

Project Name	BEVMSwap
Platform	BEVM
Language	Solidity
Codebase	Audit 1: • https://github.com/Bevmswap/bevmswap-v1.public/tree/989feef769c19ebace04d5529e00c5360c360ce0 Final Audit: • https://github.com/Bevmswap/bevmswap-v1.public/tree/2da7fad936bfbaa97037c7b2d504161ac4832cb1

1.3. Assessment Summary

Delivery Date	Mar 28, 2024
Audit Methodology	Static Analysis, Formal Verification, Manual Review

1.4. Assessment Scope

ID	File	File Hash
1	/src/interfaces/IERC20.sol	808d6a68d7088fd698f108527ce7a7cd
2	/src/interfaces/IUniswapV2Callee.sol	112295f488640c0af09f5b13b593a0d0
3	/src/interfaces/IUniswapV2ERC20.sol	4b0df0a6c79a257fa95c0c459285d8d7
4	/src/interfaces/IUniswapV2Factory.sol	9686a7c1831799c29418d922d13e63cd

ID	File	File Hash
5	/src/interfaces/IUniswapV2Pair.sol	5c37629df9544a2cd1da8f7da2a1e11c
6	/src/interfaces/IUniswapV2Router01.sol	821d31ae4f55a56897c0061721ce3d5
7	/src/interfaces/IUniswapV2Router02.sol	a05061f8208f951898d53ee4b58a8e8f
8	/src/interfaces/IWETH.sol	1723105f0310fcacdc5a70f9c76985a1
9	/src/libraries/UniswapV2Library.sol	815d17803045761dbf023681c791b46d
10	/src/libraries/UQ112x112.sol	b1cfb1755d9bb6709ecc9599108086fe
11	/src/UniswapV2ERC20.sol	ec2169eb6a8ed2c0acef0f85c78220b0
12	/src/UniswapV2Factory.sol	58e7eb48bb14037c60072391fb94d095
13	/src/UniswapV2Pair.sol	bfc1a4f13fcffc33c2275c2754171d5
14	/src/UniswapV2Router02.sol	c933a09d9b1a7a6191fc52d412edc29e

2. Checklist

2.1. Code Security

Reentrancy	DelegateCall	Integer Overflow
Input Validation	Unchecked this.call	Frozen Money
Arbitrary External Call	Unchecked Owner Transfer	Do-while Continue
Right-To-Left-Override Character	Unauthenticated Storage Access	Risk For Weak Randomness
TxOrigin	Missing Checks for Return Values	Diamond Inheritance
ThisBalance	VarType Deduction	Array Length Manipulation
Uninitialized Variable	Shadow Variable	Divide Before Multiply
Affected by Compiler Bug		

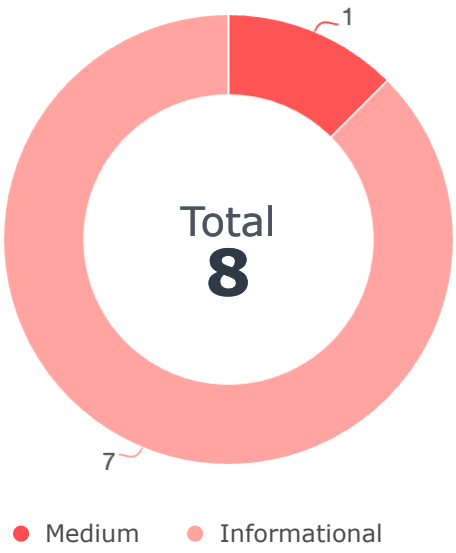
2.2. Optimization Suggestion

Compiler Version	Improper State Variable Modification
Function Visibility	Deprecated Function
Externally Controlled Variables	Code Style
Constant Specific	Event Specific
Return Value Unspecified	Inexistent Error Message
State Variable Defined Without Storage Location	Import Issue
Compare With Timestamp/Block Number/Blockhash	Constructor in Base Contract Not Implemented
Delete Struct Containing the Mapping Type	Usage of '=+'
Paths in the Modifier Not End with "_" or Revert	Non-payable Public Functions Use msg.value
Lack of SafeMath	Compiler Error/Warning
Tautology Issue	Loop Depends on Array Length
Redundant/Duplicated/Dead Code	Code Complexity/Code Inefficiency
Undeclared Resource	Optimizable Return Statement
Unused Resource	

2.3. Business Security

The Code Implementation is Consistent With Comments, Project White Papers and Other Materials
Permission Check
Address Check

3. Findings



Code Security

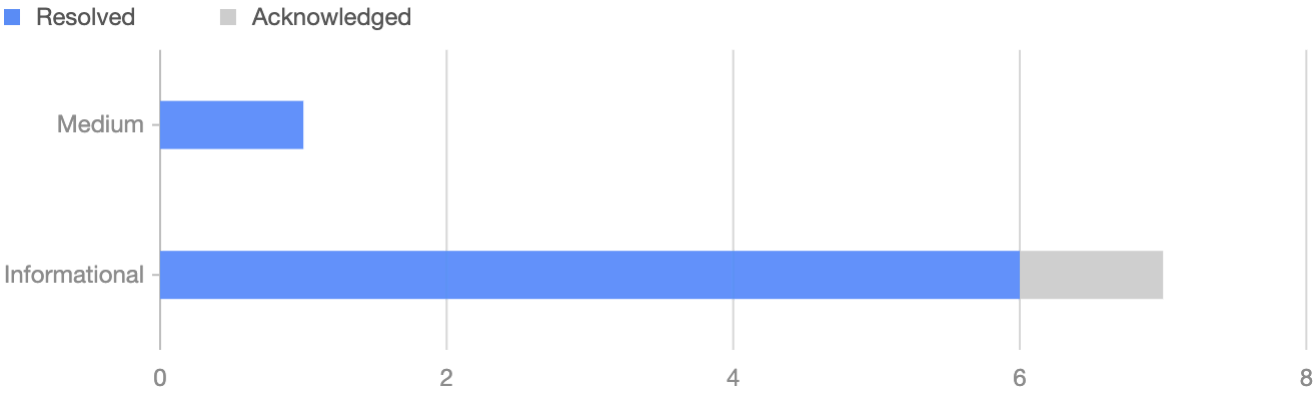
no vulnerabilities.

Optimization Suggestion

Informational: 7

Business Security

Medium: 1



ID	Title	Category	Severity	Status
M-01	Potential Inaccuracy in getAmountIn / getAmountOut Calculations	Business Security	Medium	Resolved
I-02	Inconsistent Implementation And Comments	Optimization Suggestion	Informational	Resolved
I-03	Use ++i/--i Instead of i++/i--	Optimization Suggestion	Informational	Resolved
I-04	Set the Constant to Private	Optimization Suggestion	Informational	Acknowledged
I-05	Clarify Return Value	Optimization Suggestion	Informational	Resolved
I-06	Floating Pragma	Optimization Suggestion	Informational	Resolved
I-07	Variables Can Be Declared as Immutable	Optimization Suggestion	Informational	Resolved
I-08	Use != 0 Instead of > 0 for Unsigned Integer Comparison	Optimization Suggestion	Informational	Resolved

M-01: Potential Inaccuracy in getAmountIn / getAmountOut Calculations

Medium: Business Security

File Location: /src/libraries/UniswapV2Library.sol:70,89

Description

When pre-calculating the number of tokens for swap in and out, `getAmountIn` and `getAmountOut` use a fee rate of 0.4% (i.e., `996/1000`). However, in the `UniswapV2Pair` contract, the actual fee rate used is 0.3% (i.e., `997/1000`). This may result in a mismatch between the actual token transaction volume and the expected volume calculated.

/src/libraries/UniswapV2Library.sol

```
58      // given an input amount of an asset and pair reserves, returns the
        maximum output amount of the other asset
59      function getAmountOut(
60          uint256 amountIn,
61          uint256 reserveIn,
62          uint256 reserveOut
63      )
64          internal
65          pure
66          returns (uint256 amountOut)
67      {
68          require(amountIn > 0, "UniswapV2Library:
            INSUFFICIENT_INPUT_AMOUNT");
69          require(reserveIn > 0 && reserveOut > 0, "UniswapV2Library:
            INSUFFICIENT_LIQUIDITY");
70          uint256 amountInWithFee = amountIn.mul(996);
71          uint256 numerator = amountInWithFee.mul(reserveOut);
72          uint256 denominator = reserveIn.mul(1000).add(amountInWithFee);
73          amountOut = numerator / denominator;
74      }
```

/src/libraries/UniswapV2Library.sol

```
76      // given an output amount of an asset and pair reserves, returns a
        required input amount of the other asset
77      function getAmountIn(
78          uint256 amountOut,
79          uint256 reserveIn,
80          uint256 reserveOut
81      )
82          internal
83          pure
84          returns (uint256 amountIn)
85      {
86          require(amountOut > 0, "UniswapV2Library:
            INSUFFICIENT_OUTPUT_AMOUNT");
```

```
87          require(reserveIn > 0 && reserveOut > 0, "UniswapV2Library:
            INSUFFICIENT_LIQUIDITY");
88          uint256 numerator = reserveIn.mul(amountOut).mul(1000);
89          uint256 denominator = reserveOut.sub(amountOut).mul(996);
90          amountIn = (numerator / denominator).add(1);
91      }
```

Recommendation

We recommend using a uniform fee rate in both calculations and execution.

Alleviation

Resolved in commit 2da7fad. The project team fixed the issue by unifying the fee rate in the library `UniswapV2Library` and the contract `UniswapV2Pair` to 0.4%.

I-02: Inconsistent Implementation And Comments

Informational: Optimization Suggestion

File Location: /src/UniswapV2Pair.sol:67,79,90

Description

There are two instances in the project where the code implementation does not match the comments.

First, the comment in the `_mintFee` function clearly states "if fee is on, mint liquidity equivalent to 1/6th of the growth in sqrt(k)", but in the implementation, the code calculating the `denominator` is not `rootK.mul(5).add(rootKLast)`, but `rootK.mul(1).add(rootKLast)`. The liquidity calculated from the `denominator` and `numerator` is not equivalent to 1/6th of the growth in `sqrt(k)`.

Second, in the `_update` function, when calculating `timeElapsed`, the comment explicitly states "overflow is desired". However, the contract specifying the Solidity version as 0.8.23, which includes overflow checks by default. If `blockTimestamp - blockTimestampLast` overflows, the transaction will revert directly, which is inconsistent with the description in the comment.

/src/UniswapV2Pair.sol

```
63  // update reserves and, on the first call per block, price accumulators
64  function _update(uint256 balance0, uint256 balance1, uint112 _reserve0,
    uint112 _reserve1) private {
65      require(balance0 <= type(uint112).max && balance1 <= type(uint112).
        max, "UniswapV2: OVERFLOW");
66      uint32 blockTimestamp = uint32(block.timestamp % 2 ** 32);
67      uint32 timeElapsed = blockTimestamp - blockTimestampLast; //
        overflow is desired
68      if (timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0) {
69          // * never overflows, and + overflow is desired
70          price0CumulativeLast += uint256(UQ112x112.encode(_reserve1).uqdiv
            (_reserve0)) * timeElapsed;
71          price1CumulativeLast += uint256(UQ112x112.encode(_reserve0).uqdiv
            (_reserve1)) * timeElapsed;
72      }
73      reserve0 = uint112(balance0);
74      reserve1 = uint112(balance1);
75      blockTimestampLast = blockTimestamp;
76      emit Sync(reserve0, reserve1);
77  }
```

/src/UniswapV2Pair.sol

```
79  // if fee is on, mint liquidity equivalent to 1/6th of the growth in sqrt
    (k)
80  function _mintFee(uint112 _reserve0, uint112 _reserve1) private returns
    (bool feeOn) {
81      address feeTo = IUniswapV2Factory(factory).feeTo();
82      feeOn = feeTo != address(0);
83      uint256 _kLast = kLast; // gas savings
84      if (feeOn) {
85          if (_kLast != 0) {
86              uint256 rootK = Math.sqrt(uint256(_reserve0).mul(_reserve1));
87              uint256 rootKLast = Math.sqrt(_kLast);
88              if (rootK > rootKLast) {
89                  uint256 numerator = totalSupply.mul(rootK.sub
                    (rootKLast));
90                  uint256 denominator = rootK.mul(1).add(rootKLast);
91                  uint256 liquidity = numerator / denominator;
92                  if (liquidity > 0) _mint(feeTo, liquidity);
93              }
94          }
95      } else if (_kLast != 0) {
96          kLast = 0;
97      }
98  }
```

Recommendation

It is recommended that the project team verify whether the implementation meets the expected logic.

Alleviation

Resolved in commit 2da7fad. The project team fixed the issue by modifying the comment in the `_mintFee` function to "if fee is on, mint liquidity equivalent to 1/2th of the growth in sqrt(k)" and deleting the comment "overflow is desired" in the function `_update`.

I-03: Use ++i/--i Instead of i++/i--

Informational: Optimization Suggestion

File Location:
/src/UniswapV2Router02.sol:259,409
/src/libraries/UniswapV2Library.sol:106,125

Description

Compared with i++, ++i can save about 5 gas per use. Compared with i--, --i can save about 3 gas per use in for loop.

/src/UniswapV2Router02.sol

```
257     // requires the initial amount to have already been sent to the
        first pair
258     function _swap(uint256[] memory amounts, address[] memory path,
        address _to) internal virtual {
259         for (uint256 i; i < path.length - 1; i++) {
260             (address input, address output) = (path[i], path[i + 1]);
261             (address token0,) = UniswapV2Library.sortTokens(input,
                output);
```

/src/UniswapV2Router02.sol

```
407     // requires the initial amount to have already been sent to the
        first pair
408     function _swapSupportingFeeOnTransferTokens(address[] memory path,
        address _to) internal virtual {
409         for (uint256 i; i < path.length - 1; i++) {
410             (address input, address output) = (path[i], path[i + 1]);
411             (address token0,) = UniswapV2Library.sortTokens(input,
                output);
```

/src/libraries/UniswapV2Library.sol

```
104         amounts = new uint256[](path.length);
105         amounts[0] = amountIn;
106         for (uint256 i; i < path.length - 1; i++) {
107             (uint256 reserveIn, uint256 reserveOut) = getReserves
                (factory, path[i], path[i + 1]);
108             amounts[i + 1] = getAmountOut(amounts[i], reserveIn,
                reserveOut);
```

/src/libraries/UniswapV2Library.sol

```
123         amounts = new uint256[](path.length);
124         amounts[amounts.length - 1] = amountOut;
125         for (uint256 i = path.length - 1; i > 0; i--) {
126             (uint256 reserveIn, uint256 reserveOut) = getReserves
            (factory, path[i - 1], path[i]);
127             amounts[i - 1] = getAmountIn(amounts[i], reserveIn,
            reserveOut);
```

Recommendation

It is recommended to use ++i/--i instead of i++/i-- in for loop.

Alleviation

Resolved in commit 2da7fad.

I-04: Set the Constant to Private

Informational: Optimization Suggestion

File Location:
/src/UniswapV2ERC20.sol:19
/src/UniswapV2Pair.sol:17

Description

For constants, if the visibility is set to public, the compiler will automatically generate a getter function for it, which will consume more gas during deployment.

/src/UniswapV2ERC20.sol

```
17      bytes32 public DOMAIN_SEPARATOR;
18      // keccak256("Permit(address owner,address spender,uint256 value,
        uint256 nonce,uint256 deadline)");
19      bytes32 public constant PERMIT_TYPEHASH =
        0x6e71edae12b1b97f4d1f60370fef10105fa2faae0126114a169c64845d6126c9;
20      mapping(address => uint256) public nonces;
21
```

/src/UniswapV2Pair.sol

```
15      using UQ112x112 for uint224;
16
17      uint256 public constant MINIMUM_LIQUIDITY = 10 ** 3;
18      bytes4 private constant SELECTOR = bytes4(keccak256(bytes("transfer
        (address,uint256)")));
19
```

Recommendation

It is recommended to set the visibility of constants to private instead of public.

Alleviation

Acknowledged. This issue will not be modified because there may be external APIs or contracts that could rely on the public nature of the variables.

I-05: Clarify Return Value

Informational: Optimization Suggestion

File Location: /src/UniswapV2Router02.sol:512,526,540,553,566

Description

The returned variable is specified in the function signature, but it still calls the return statement to return a local variable defined in the function body or state variable. It is necessary to clarify whether the returned value meets expectations.

/src/UniswapV2Router02.sol

```
510             returns (uint256 amountB)
511         {
512             return UniswapV2Library.quote(amountA, reserveA, reserveB);
513         }
514
```

/src/UniswapV2Router02.sol

```
524             returns (uint256 amountOut)
525         {
526             return UniswapV2Library.getAmountOut(amountIn, reserveIn,
                reserveOut);
527         }
528
```

/src/UniswapV2Router02.sol

```
538             returns (uint256 amountIn)
539         {
540             return UniswapV2Library.getAmountIn(amountOut, reserveIn,
                reserveOut);
541         }
542
```

/src/UniswapV2Router02.sol

```
551             returns (uint256[] memory amounts)
552         {
553             return UniswapV2Library.getAmountsOut(factory, amountIn, path);
554         }
555
```

/src/UniswapV2Router02.sol

```
564             returns (uint256[] memory amounts)
565         {
566             return UniswapV2Library.getAmountsIn(factory, amountOut, path);
567         }
568     }
```

Recommendation

It is recommended to be clear whether the returned value is as expected, and only use one way to return the value.

Alleviation

Resolved in commit 2da7fad.

I-06: Floating Pragma

Informational: Optimization Suggestion

File Location:
/src/interfaces/IERC20.sol:1
/src/interfaces/IUniswapV2Callee.sol:1
/src/interfaces/IUniswapV2ERC20.sol:1
/src/interfaces/IUniswapV2Factory.sol:1
/src/interfaces/IUniswapV2Pair.sol:1
/src/interfaces/IUniswapV2Router01.sol:1
/src/interfaces/IUniswapV2Router02.sol:1
/src/interfaces/IWETH.sol:1

Description

Contracts should be deployed with fixed compiler version which has been tested thoroughly or make sure to lock the contract compiler version in the project configuration. Locked compiler version ensures that contracts will not be compiled by untested compiler version.

/src/interfaces/IERC20.sol

```
1  pragma solidity >=0.8.23;
2
3  interface IERC20 {
```

/src/interfaces/IUniswapV2Callee.sol

```
1  pragma solidity >=0.8.23;
2
3  interface IUniswapV2Callee {
```

/src/interfaces/IUniswapV2ERC20.sol

```
1  pragma solidity >=0.8.23;
2
3  import { IERC20 } from "./IERC20.sol";
```

/src/interfaces/IUniswapV2Factory.sol

```
1  pragma solidity >=0.8.23;
2
3  interface IUniswapV2Factory {
```

/src/interfaces/IUniswapV2Pair.sol

```
1  pragma solidity >=0.8.23;
2
3  import { IUniswapV2ERC20 } from "./IUniswapV2ERC20.sol";
```

/src/interfaces/IUniswapV2Router01.sol

```
1  pragma solidity >=0.8.23;
2
3  interface IUniswapV2Router01 {
```

/src/interfaces/IUniswapV2Router02.sol

```
1  pragma solidity >=0.8.23;
2
3  import { IUniswapV2Router01 } from "./IUniswapV2Router01.sol";
```

/src/interfaces/IWETH.sol

```
1  pragma solidity >=0.8.23;
2
3  interface IWETH {
```

Recommendation

Use a fixed compiler version, and consider whether the bugs in the selected compiler version (https://github.com/ethereum/solidity/releases) will affect the contract.

Alleviation

Resolved in commit 2da7fad.

I-07: Variables Can Be Declared as Immutable

Informational: Optimization Suggestion

File Location:
/src/UniswapV2ERC20.sol:17
/src/UniswapV2Pair.sol:20

Description

The solidity compiler of version 0.6.5 introduces immutable to modify state variables that are only modified in the constructor. Using immutable can save gas.

/src/UniswapV2ERC20.sol

```
15      mapping(address => mapping(address => uint256)) public allowance;
16
17      bytes32 public DOMAIN_SEPARATOR;
18      // keccak256("Permit(address owner,address spender,uint256 value,
        uint256 nonce,uint256 deadline)");
19      bytes32 public constant PERMIT_TYPEHASH =
        0x6e71edae12b1b97f4d1f60370fef10105fa2faae0126114a169c64845d6126c9;
```

/src/UniswapV2Pair.sol

```
18      bytes4 private constant SELECTOR = bytes4(keccak256(bytes("transfer
        (address,uint256)")));
19
20      address public factory;
21      address public token0;
22      address public token1;
```

Recommendation

For contracts compiled with compiler of versions 0.6.5 and above, if the state variable is only modified in the constructor, it is recommended to modify the variable with immutable to save gas.

Alleviation

Resolved in commit 2da7fad.

I-08: Use != 0 Instead of > 0 for Unsigned Integer Comparison

Informational: Optimization Suggestion

File Location:
/src/UniswapV2Pair.sol:153,164,165
/src/libraries/UniswapV2Library.sol:53,54,68,69,86,87

Description

For unsigned integers, use !=0 for comparison, which consumes less gas than >0. When compiler optimization is turned off, about 3 gas can be saved. When compiler optimization is turned on, no gas can be saved.

/src/UniswapV2Pair.sol

```
151     // this low-level function should be called from a contract which
        performs important safety checks
152     function swap(uint256 amount0Out, uint256 amount1Out, address to,
        bytes calldata data) external lock {
153         require(amount0Out > 0 || amount1Out > 0, "UniswapV2:
            INSUFFICIENT_OUTPUT_AMOUNT");
154         (uint112 _reserve0, uint112 _reserve1,) = getReserves(); // gas
            savings
155         require(amount0Out < _reserve0 && amount1Out < _reserve1,
            "UniswapV2: INSUFFICIENT_LIQUIDITY");
```

/src/UniswapV2Pair.sol

```
162             address _token1 = token1;
163             require(to != _token0 && to != _token1, "UniswapV2:
                INVALID_TO");
164             if (amount0Out > 0) _safeTransfer(_token0, to,
                amount0Out); // optimistically transfer tokens
165             if (amount1Out > 0) _safeTransfer(_token1, to,
                amount1Out); // optimistically transfer tokens
166             if (data.length > 0) IUniswapV2Callee(to).uniswapV2Call(msg.
                sender, amount0Out, amount1Out, data);
```

/src/UniswapV2Pair.sol

```
163             require(to != _token0 && to != _token1, "UniswapV2:
                INVALID_TO");
164             if (amount0Out > 0) _safeTransfer(_token0, to,
                amount0Out); // optimistically transfer tokens
165             if (amount1Out > 0) _safeTransfer(_token1, to,
                amount1Out); // optimistically transfer tokens
166             if (data.length > 0) IUniswapV2Callee(to).uniswapV2Call(msg.
                sender, amount0Out, amount1Out, data);
167             balance0 = IERC20(_token0).balanceOf(address(this));
```

/src/libraries/UniswapV2Library.sol

```
51      // given some amount of an asset and pair reserves, returns an
        equivalent amount of the other asset
52      function quote(uint256 amountA, uint256 reserveA, uint256 reserveB)
        internal pure returns (uint256 amountB) {
53          require(amountA > 0, "UniswapV2Library: INSUFFICIENT_AMOUNT");
54          require(reserveA > 0 && reserveB > 0, "UniswapV2Library:
            INSUFFICIENT_LIQUIDITY");
55          amountB = amountA.mul(reserveB) / reserveA;
```

/src/libraries/UniswapV2Library.sol

```
52      function quote(uint256 amountA, uint256 reserveA, uint256 reserveB)
        internal pure returns (uint256 amountB) {
53          require(amountA > 0, "UniswapV2Library: INSUFFICIENT_AMOUNT");
54          require(reserveA > 0 && reserveB > 0, "UniswapV2Library:
            INSUFFICIENT_LIQUIDITY");
55          amountB = amountA.mul(reserveB) / reserveA;
56      }
```

/src/libraries/UniswapV2Library.sol

```
66          returns (uint256 amountOut)
67      {
68          require(amountIn > 0, "UniswapV2Library:
            INSUFFICIENT_INPUT_AMOUNT");
69          require(reserveIn > 0 && reserveOut > 0, "UniswapV2Library:
            INSUFFICIENT_LIQUIDITY");
70          uint256 amountInWithFee = amountIn.mul(996);
```

/src/libraries/UniswapV2Library.sol

```
67      {
68          require(amountIn > 0, "UniswapV2Library:
            INSUFFICIENT_INPUT_AMOUNT");
69          require(reserveIn > 0 && reserveOut > 0, "UniswapV2Library:
            INSUFFICIENT_LIQUIDITY");
70          uint256 amountInWithFee = amountIn.mul(996);
71          uint256 numerator = amountInWithFee.mul(reserveOut);
```

/src/libraries/UniswapV2Library.sol

```
84          returns (uint256 amountIn)
85      {
86          require(amountOut > 0, "UniswapV2Library:
            INSUFFICIENT_OUTPUT_AMOUNT");
87          require(reserveIn > 0 && reserveOut > 0, "UniswapV2Library:
            INSUFFICIENT_LIQUIDITY");
88          uint256 numerator = reserveIn.mul(amountOut).mul(1000);
```

/src/libraries/UniswapV2Library.sol

```
85      {
86          require(amountOut > 0, "UniswapV2Library:
            INSUFFICIENT_OUTPUT_AMOUNT");
87          require(reserveIn > 0 && reserveOut > 0, "UniswapV2Library:
            INSUFFICIENT_LIQUIDITY");
88          uint256 numerator = reserveIn.mul(amountOut).mul(1000);
89          uint256 denominator = reserveOut.sub(amountOut).mul(996);
```

Recommendation

For unsigned integers, it is recommended to use !=0 instead of >0 for comparison.

Alleviation

Resolved in commit 2da7fad.

4. Disclaimer

No description, statement, recommendation or conclusion in this report shall be construed as endorsement, affirmation or confirmation of the project. The security assessment is limited to the scope of work as stipulated in the Statement of Work.

This report is prepared in response to source code, and based on the attacks and vulnerabilities in the source code that already existed or occurred before the date of this report, excluding any new attacks or vulnerabilities that exist or occur after the date of this report. The security assessment are solely based on the documents and materials provided by the customer, and the customer represents and warrants documents and materials are true, accurate and complete.

CONSULTANT DOES NOT MAKE AND HEREBY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, REGARDING THE SERVICES, DELIVERABLES, OR ANY OTHER MATTER PERTAINING TO THIS REPORT.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR AND HEREBY DISCLAIMS MERCHANTABILITY, FITNESS FOR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-APPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, SATISFACTORY QUALITY, ACCURACY, QUALITY, COMPLETENESS, TIMELINESS, RESPONSIVENESS, OR PRODUCTIVITY OF THE SERVICES OR DELIVERABLES.

CONSULTANT EXCLUDES ANY WARRANTY THAT THE SERVICES AND DELIVERABLES WILL BE UNINTERRUPTED, ERROR FREE, FREE OF SECURITY DEFECTS OR HARMFUL COMPONENTS, REVEAL ALL SECURITY VULNERABILITIES, OR THAT ANY DATA WILL NOT BE LOST OR CORRUPTED.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR (A) ANY REPRESENTATIONS MADE BY ANY PERSON REGARDING THE SUFFICIENCY OR SUITABILITY OF SERVICES AND DELIVERABLES IN ANY ACTUAL APPLICATION, OR (B) WHETHER ANY SUCH USE WOULD VIOLATE OR INFRINGE THE APPLICABLE LAWS, OR (C) REVIEWING THE CUSTOMER MATERIALS FOR ACCURACY.

5. Appendix

5.1 Visibility

Contract	FuncName	Visibility	Mutability	Modifiers
UniswapV2Router02	_CTOR_	public	Y	
UniswapV2Router02	receive	external	N	
UniswapV2Router02	_addLiquidity	internal	N	
UniswapV2Router02	addLiquidity	external	N	ensure
UniswapV2Router02	addLiquidityETH	external	N	ensure
UniswapV2Router02	removeLiquidity	public	N	ensure
UniswapV2Router02	removeLiquidityETH	public	N	ensure
UniswapV2Router02	removeLiquidityWithPermit	external	N	
UniswapV2Router02	removeLiquidityETHWithPermit	external	N	
UniswapV2Router02	removeLiquidityETHSupportingFeeOnTransferTokens	public	N	ensure
UniswapV2Router02	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	external	N	
UniswapV2Router02	_swap	internal	N	
UniswapV2Router02	swapExactTokensForTokens	external	N	ensure
UniswapV2Router02	swapTokensForExactTokens	external	N	ensure

Contract	FuncName	Visibility	Mutability	Modifiers
UniswapV2Router02	swapExactETHForTokens	external	N	ensure
UniswapV2Router02	swapTokensForExactETH	external	N	ensure
UniswapV2Router02	swapExactTokensForETH	external	N	ensure
UniswapV2Router02	swapETHForExactTokens	external	N	ensure
UniswapV2Router02	_swapSupportingFeeOnTransferTokens	internal	N	
UniswapV2Router02	swapExactTokensForTokensSupportingFeeOnTransferTokens	external	N	ensure
UniswapV2Router02	swapExactETHForTokensSupportingFeeOnTransferTokens	external	N	ensure
UniswapV2Router02	swapExactTokensForETHSupportingFeeOnTransferTokens	external	N	ensure
UniswapV2Router02	quote	public	N	
UniswapV2Router02	getAmountOut	public	N	
UniswapV2Router02	getAmountIn	public	N	
UniswapV2Router02	getAmountsOut	public	N	
UniswapV2Router02	getAmountsIn	public	N	
UniswapV2ERC20	_CTOR_	public	Y	
UniswapV2ERC20	_mint	internal	N	
UniswapV2ERC20	_burn	internal	N	

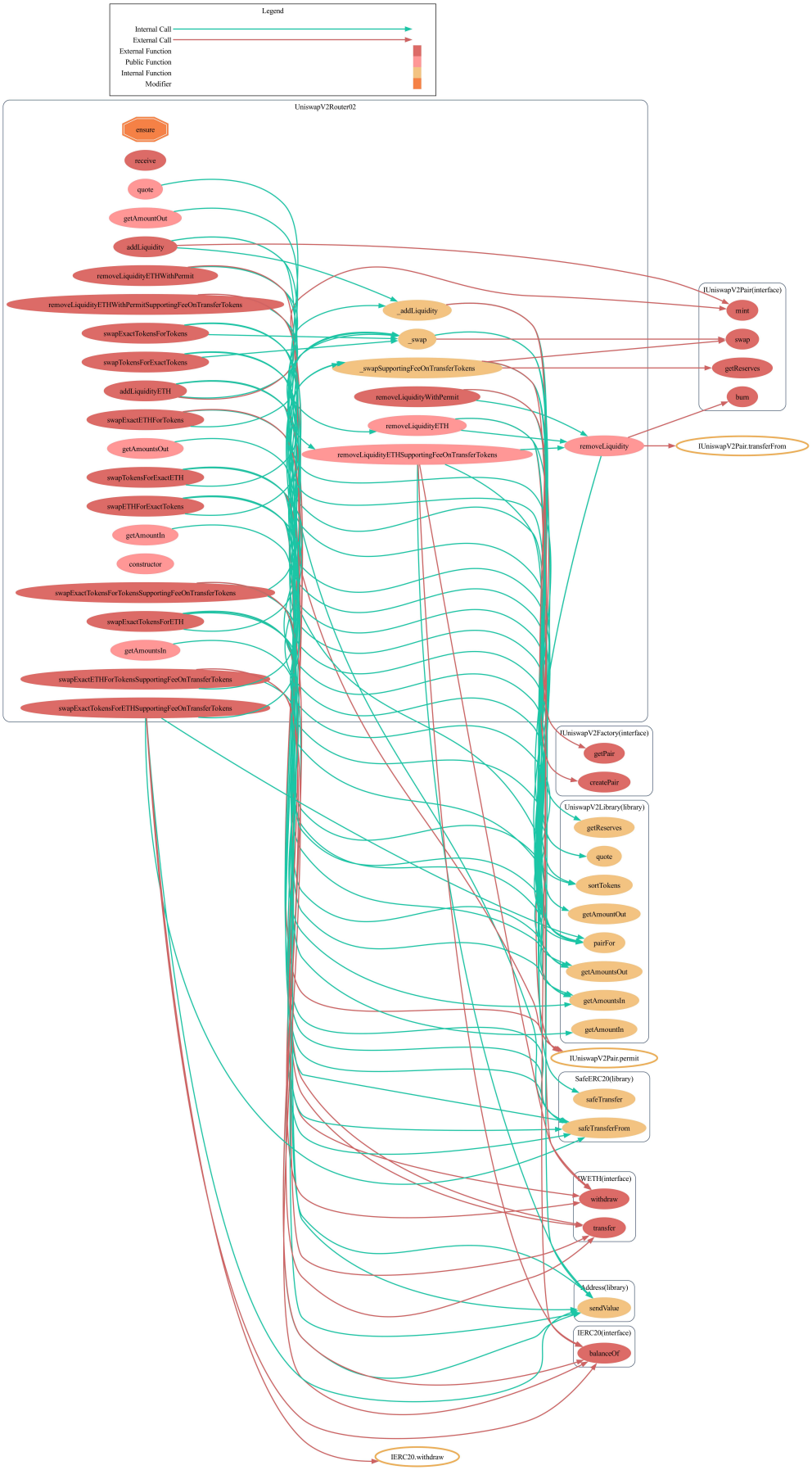
Contract	FuncName	Visibility	Mutability	Modifiers
UniswapV2ERC20	_approve	private	N	
UniswapV2ERC20	_transfer	private	N	
UniswapV2ERC20	approve	external	Y	
UniswapV2ERC20	transfer	external	Y	
UniswapV2ERC20	transferFrom	external	Y	
UniswapV2ERC20	permit	external	Y	
UniswapV2Pair	getReserves	public	N	
UniswapV2Pair	_safeTransfer	private	N	
UniswapV2Pair	_CTOR_	public	Y	
UniswapV2Pair	initialize	external	Y	
UniswapV2Pair	_update	private	N	
UniswapV2Pair	_mintFee	private	N	
UniswapV2Pair	mint	external	Y	lock
UniswapV2Pair	burn	external	Y	lock
UniswapV2Pair	swap	external	Y	lock
UniswapV2Pair	skim	external	Y	lock
UniswapV2Pair	sync	external	Y	lock
UniswapV2Factory	_CTOR_	public	Y	
UniswapV2Factory	allPairsLength	external	N	
UniswapV2Factory	createPair	external	Y	
UniswapV2Factory	setFeeTo	external	Y	

Contract	FuncName	Visibility	Mutability	Modifiers
UniswapV2Factory	setFeeToSetter	external	Y	

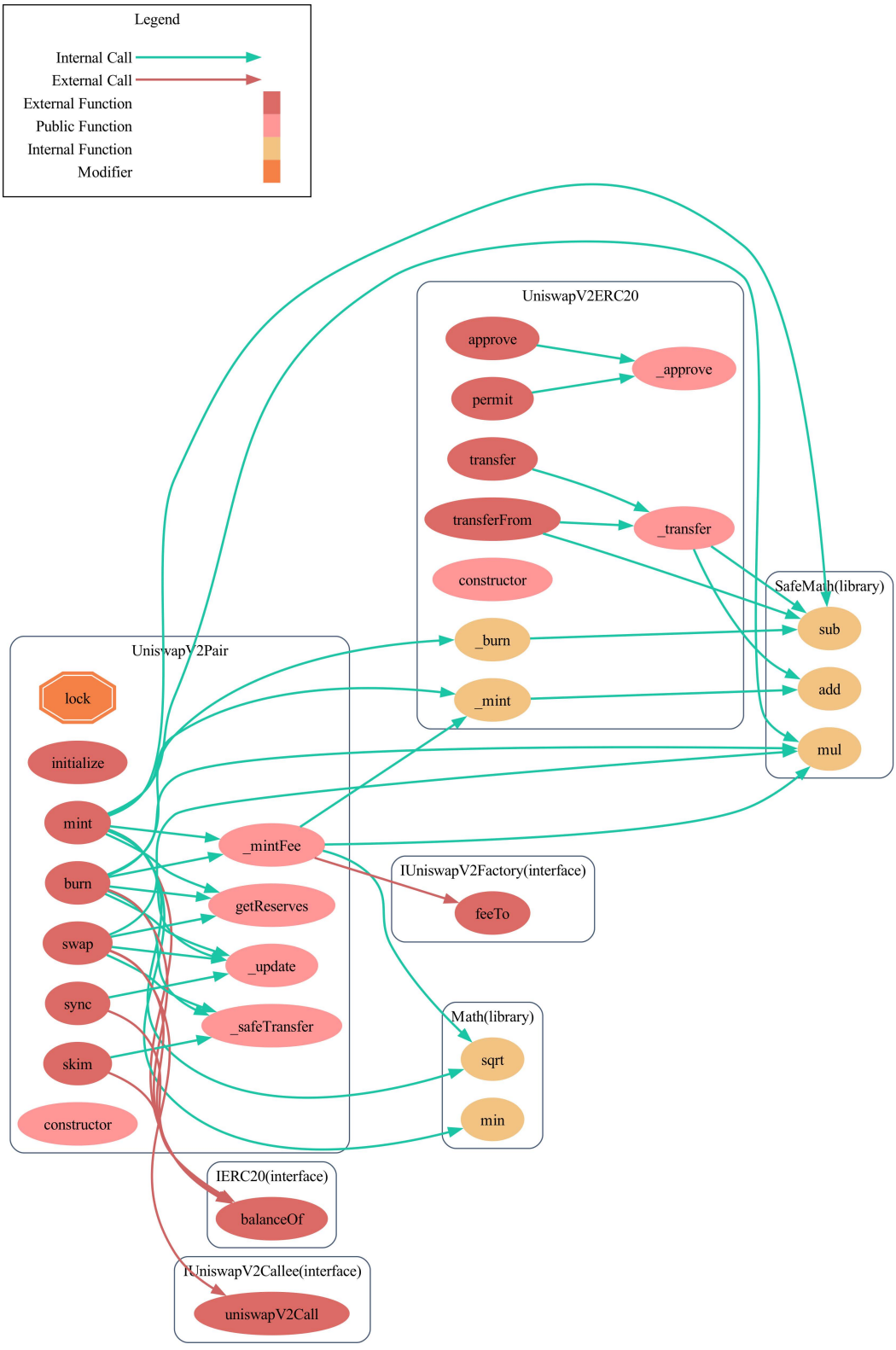
5. Appendix

5.2 Call Graph

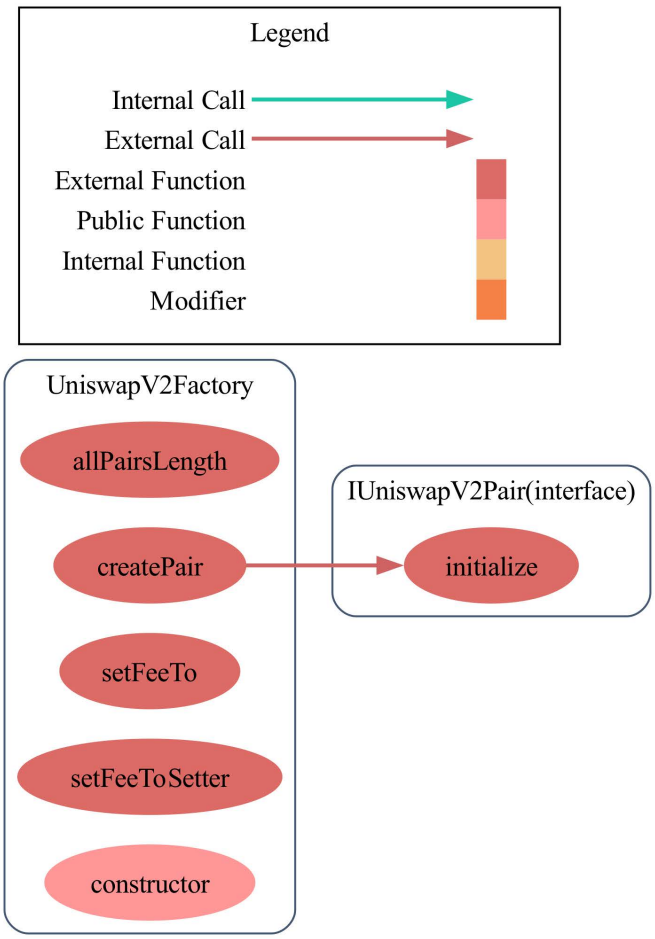
UniswapV2Router02



UniswapV2Pair



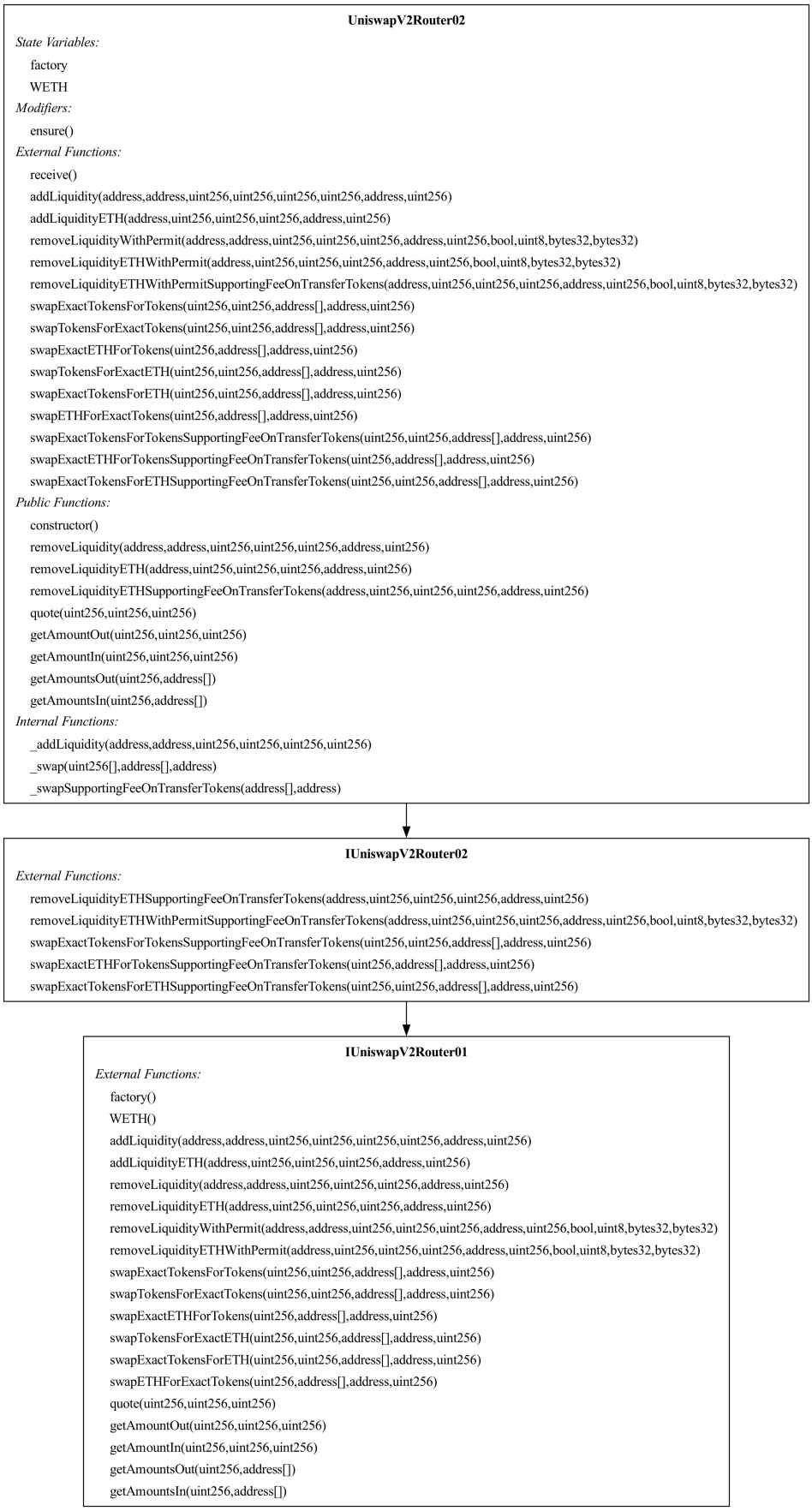
UniswapV2Factory



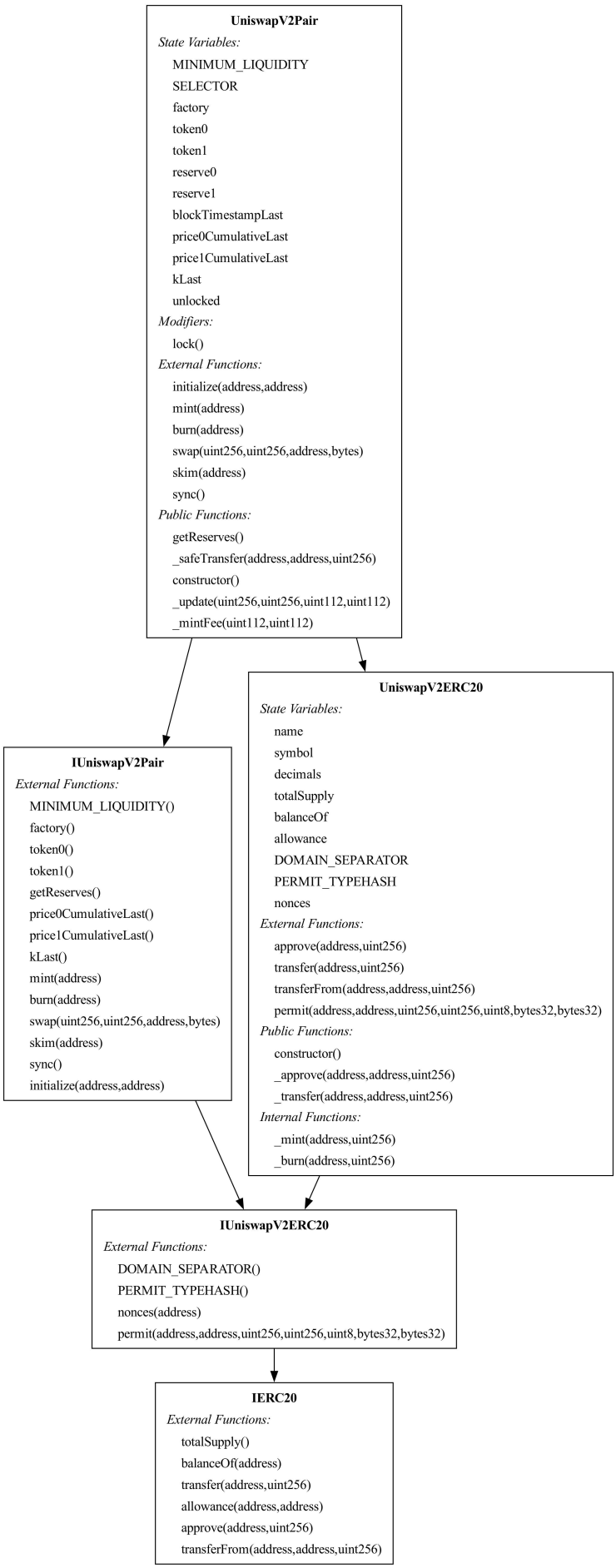
5. Appendix

5.3 Inheritance Graph

UniswapV2Router02



UniswapV2Pair



UniswapV2Factory